Avaliação de Desempenho de Computação Móvel na Borda Usando Redes de Petri Estocásticas

Brena Santos[∓], Daniel Carvalho[∓], Iure Fé[⋆] e Francisco Airton Silva[∓]

[1] ∓ Universidade Federal do Piauí (UFPI)
Núcleo de Pesquisas Aplicadas a Sistemas Distribuídos (PASID)
⋆ 3º BEC - Exército Brasileiro
E-mail:faps@ufpi.edu.br

Abstract. *Mobile Edge Computing (MEC) has emerged as an alternative to reduce network latency that has leaded to the data flow processing closer to the user. However, the server machines resource capacity can directly influence MEC performance. This paper proposes a Stochastic Petri Net (SPN) model to such a scenario and analyzes its performance, considering several parameters that can directly affect the Mean Response Time (MRT) and Utilization Level. We also present numerical analysis that serve as a practical guide to assist computer infrastructure managers to design their architectures, finding the trade-off between MRT and utilization.*

Resumo. *A Computação Móvel na Borda (MEC) surgiu como uma alternativa para diminuir a latência da rede, fazendo com que o processamento do fluxo de dados ocorresse mais próximo dos usuários mobile. No entanto, a configuração dos servidores pode influenciar diretamente no desempenho da arquitetura MEC. Este artigo propõe um modelo de Rede de Petri Estocástica (SPN) para modelar tal cenário e analisar seu desempenho, considerando diversos parâmetros que podem afetar diretamente no Tempo Médio de Resposta (MRT) e no Nível de Utilização. Apresentamos também análises numéricas com valores de entrada reais que servem como um guia prático para auxiliar administradores de infraestruturas computacionais a adequar suas arquiteturas, encontrando o trade-off entre o MRT e o nível de utilização.*

1. Introdução

De acordo com o relatório apresentado pela Cisco Visual Networking Index, o número de dispositivos móveis chegará a 11,6 bilhões em 2020 [Jung 2016]. A tendência de aumento no uso de dispositivos móveis é fundamentalmente impulsionada pelo aumento de usuários móveis e pelo desenvolvimento de aplicativos cada vez mais interativos e utilitários [Kitanov et al. 2016] [Beck et al. 2014]. O crescimento do uso de smartphones e tablets teve um efeito extremo sobre as redes móveis trazendo grandes desafios para as companhias de telecomunicações [Cau et al. 2016].

As redes celulares devem suportar a todo custo a baixa capacidade de armazenamento, o alto consumo de energia, a baixa largura de banda e altas latências [Orsini et al. 2015]. Com o objetivo de propor arquiteturas cada vez mais otimizadas para este contexto, surgiu a computação móvel na borda, do inglês *Mobile Edge Computing* (MEC). O principal objetivo da MEC é implantar os recursos ainda mais perto dos usuários — na borda da rede. Assim, a computação e armazenamento são executados mais

próximos do local de origem, sempre na casa dos milissegundos [Jararweh et al. 2016]. Por ser uma área ainda recente e com grandes desafios, algumas lacunas de pesquisa ainda devem ser exploradas, como é o caso da avaliação de desempenho de arquiteturas MEC.

Os estudos mais recentes sobre avaliação de desempenho em MEC apresentam limitações quanto ao número de fatores analisados de forma conjunta devido à alta complexidade e alto custo. Neste caso, modelos analíticos, em especial Redes de Petri Estocásticas (SPNs) se tornam uma alternativa adequada para avaliação de desempenho deste tipo de sistemas. Redes de Petri Estocásticas permitem modelar e avaliar desempenho de sistemas envolvendo concorrência, não-determinismo e sincronização.

Este artigo apresenta um modelo SPN para modelar uma arquitetura MEC onde os recursos de um único servidor são paralelizados com microsserviços sendo executados em múltiplos contêineres. O modelo proposto neste trabalho permite avaliar o *trade-off* entre tempo médio de execução (MRT) e utilização de recursos. Em resumo, as principais contribuições deste artigo são:

- um modelo analítico, que é uma ferramenta útil para administradores de sistemas avaliarem o desempenho de arquiteturas MEC, antes mesmo de sua implantação;
- um conjunto de análises numéricas com dados reais que provê um guia prático para análise de desempenho em arquiteturas MEC.

2. Arquitetura e Modelo

A Figura 1 ilustra a arquitetura que propomos modelar e analisar seu desempenho. Tal arquitetura base é bastante adotada para descrever a infraestrutura MEC em diversos trabalhos [Trinh and Yao 2017, Yuanzhe and Shangguang 2018, Guangshun Li and Junhua Wu 2018] com um servidor MEC visando processar fluxos de dados recebidos. Esses dados são gerados e enviados por aplicativos executados em dispositivos móveis, por exemplo, dados de aplicações de monitoramento da saúde do usuário, dados para a renderização de jogos, etc. No contexto deste trabalho focamos em aplicações com alto nível de interatividade com o usuário, incluindo periféricos como smartphones ou tablets.

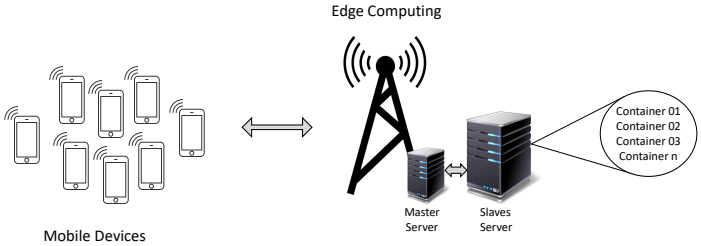


Figura 1. Arquitetura MEC Base Adotada na Avaliação de Desempenho

Na camada da *edge computing* temos um servidor *master* e um servidor com os nós *slaves* (que farão o processamento em si). O servidor *master* é responsável por receber as requisições dos dispositivos móveis e distribuí-los entre os nós slaves. Os nós *slaves* são microsserviços que são executados em contêineres. Neste trabalho, cada contêiner é configurado para executar em um núcleo do servidor. Portanto, se existirem 16 núcleos, serão executados 16 contêineres.

A Figura 2 apresenta nosso modelo SPN, composto de duas macro partes:

1. *Admission* que trata da geração das requisições;
2. *Edge* composto pelo servidor master e o servidor com nós slaves. O servidor master recebe dados e os distribui entre os nós slaves, que por fim retornam os resultados aos clientes;

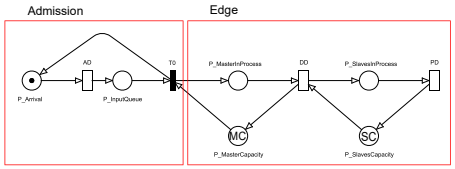


Figura 2. Modelo SPN

A Tabela 1 apresenta todos os elementos do modelo extensivamente.

Tabela 1. Descrição dos Elementos do Modelo

Tipo	Elemento	Descrição
Lugares	P_Arrival	Espera por novas requisições
	P_InputQueue	Espera pela disponibilidade da fila
	P_MasterInProcess	Trabalhos na fila do servidor master
	P_MasterCapacity	Capacidade do servidor master
	P_SlavesInProcess	Requisições na fila dos nós slaves da borda
	P_SlavesCapacity	Capacidade da borda
Transições Temporizadas	AD	Tempo de chegada entre requisições *(Single Server)*
	DD	Tempo para o master distribuir as requisições entre os slaves *(Infinite Server)*
	PD	Tempo gasto para processar o trabalho no nó slave *(Infinite Server)*
Marcações dos Lugares	MC	Capacidade máxima da fila de entrada do master
	SC	Capacidade de nós slaves disponíveis para processamento

O tempo médio de resposta (*MRT*) pode ser obtido a partir da Lei de Little [Little 1961]. Considerando a transição para tempo entre gerações do modelo, temos que $ARR = \frac{1}{AD}$. No modelo proposto pressupomos que o tempo entre chegadas real não necessariamente é o tempo entre chegadas efetivo, pois alguns trabalhos podem se perder, ou por termos filas finitas, serem descartados. Então, como recomendado pelo autor Jain (1990), nós subtraímos a probabilidade de descartes representado pela variável $N\_Discard$. $P(Lugar = n)$ calcula a probabilidade de existirem n tokens em "Lugar". Por fim, nós calculamos também a probabilidade de utilização dos recursos.

As Equações correspondentes às métricas utilizadas foram definidas da seguinte maneira: a Equação equivalente à Lei de Little para MRT foi $MRT = \frac{RequestsInProgress}{ARR \times (1 - N\_Discard)}$; a Equação para RequestsInProcess foi definida como $RequestsInProcess = Esp(P\_MasterInProcess) + Esp(P\_SlavesInProcess)$; a Equação que define N_Discard foi $N\_Discard = P((P\_InputQueue = 1) \wedge (P\_MasterCapacity = 0) \wedge (P\_SlavesCapacity = 0)$; a Equação para U_Master foi $U\_Master = \frac{Esp(P\_MasterInProcess)}{MC}$; e a Equação para U_Slaves foi $U\_Slaves = \frac{Esp(P\_MasterInProcess)}{SC}$.

3. Análises Numéricas

Esta seção apresenta duas análises numéricas a partir da resolução do modelo SPN proposto visando analisar MRT e Utilização. No trabalho de [Gopika Premsankar and Taleb 2018] (que iremos chamar de "trabalho de referência") os autores avaliaram uma arquitetura MEC com um único dispositivo móvel como cliente através de experimentos em um ambiente real com execução de serviços em contêineres. O trabalho de referência avaliou um jogo 3D chamado Neverball onde o jogador deve inclinar o piso para controlar a bola fazendo com que ela colete moedas e alcance um ponto de saída antes que o tempo acabe. Extraímos do trabalho de referência os parâmetros de entrada do modelo. Portanto, nosso estudo evolui o trabalho dos autores pois nós agora

analisamos numericamente cenários com múltiplos usuários considerando o jogo com uma resolução de 800x600 pixels. O parâmetro adotado do trabalho de referência correspondendo ao tempo de processamento (PD) de uma requisição foi de 24ms. Adotamos o tempo de 5ms para a distribuição das requisições (transição DD). Neste estudo estabelecemos que o servidor master tem uma restrição de atender no máximo 40 requisições paralelamente, ou seja setamos a marcação MC com valor 40.

O modelo permite uma ampla variedade de parametrizações. Na presente análise variamos dois parâmetros, o intervalo de tempo entre chegadas de requisições (AD) e a capacidade de recursos do servidor com nós slaves (SC). O valor de AD foi variado entre 1ms até 10ms com acréscimos de 0.5ms. A variável SC foi configurada com três possibilidades (8, 16 e 32), correspondendo a opções sobre a quantidade de núcleos de um servidor. Obviamente todos estes parâmetros poderiam ser variados de outras formas, por exemplo, o número de nós slaves poderiam não estar atrelados a número de núcleos, podendo adotar por exemplo SC na casa dos milhares. Adotando os parâmetros acima mencionados, a seguir apresentamos os resultados considerando as métricas MRT, descarte, utilização do servidor master, e utilização do servidor com nós slaves.

A Figura 3 apresenta os resultados para o MRT. A princípio espera-se que quanto maior o intervalo de tempo entre chegadas (AD) menor seja o valor de MRT, pois o sistema estará mais ocioso com menos chegadas de requisições. Espera-se também que quanto maior for a quantidade de recursos slaves (SC) menor será o valor de MRT pois se aumentará a disponibilidade de recursos, aumentando-se o nível de paralelismo. Estes dois comportamentos são facilmente observados nos resultados quando os valores de descartes de trabalhos são desprezíveis no modelo, para esses intervalos o MRT decresce até o tempo mínimo para executar trabalhos sem enfileiramento no sistema.

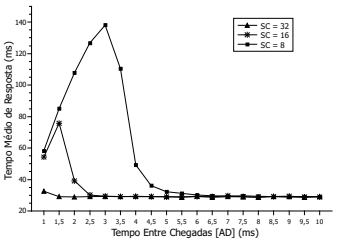


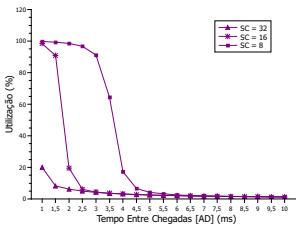
Figura 3. MRT

No entanto, para os casos que o descarte é presente no sistema, observamos o aumento do MRT até um pico para que ele decresça, este comportamento é decorrente do valor limitante de recursos no sistema, onde parte dos trabalhos entrantes são descartados quando não há mais recursos, limitando a variação do MRT ao tempo entre chegadas, como pode ser deduzido da lei de Little [Jain 1990], portanto o tempo médio entre saídas irá aumentar junto com o tempo médio entre chegadas até o pico, para SC = 8 foi em AD = 1.5ms e para SC = 16 foi de AD = 3.0ms. Ao atingir esses tempos entre chegadas, a quantidade de trabalhos dentro do sistema começa a reduzir drasticamente, reduzindo o MRT mesmo com o aumento do AD. É importante ressaltar que consideramos no MRT o tempo entre chegadas efetivo, isto é, ajustando o seu valor com a probabilidade de descarte.

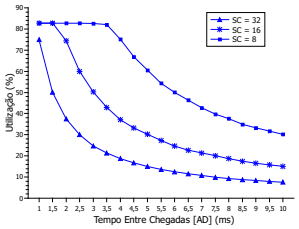
O valor de MRT para SC = 32 é extremamente baixo, mesmo para um intervalo entre chegadas de 1ms. Comparando SC = 16 e SC = 32, temos que a partir de 2.5ms os MRTs se igualam, e a partir de 5.5ms a configuração SC = 8 também apresenta o mesmo resultado médio. Portanto, caso o contexto real apresentasse um AD de 5.5ms, um servidor de 8 núcleos atingiria o mesmo desempenho do que servidores mais potentes, portanto nosso trabalho pode auxiliar no dimensionamento adequado da escolha de ser-

vidores, identificando a melhor opção de desempenho e custo para uma carga de trabalho esperada.

A Figura 4(a) apresenta o nível de utilização do servidor master. O servidor master é o primeiro componente que a requisição atinge ao entrar na camada MEC. Para as configurações SC = 8 e SC = 16, o nível de utilização gira em torno de 100% nos menores valores de AD, posteriormente este valor cai. Para SC = 32, mesmo com AD = 1.0ms, o valor de utilização chega a apenas 20%. A partir de AD = 5.5ms as três configurações possuem valores semelhantes e próximos a 0%. O administrador do sistema deve considerar o nível desejado de ociosidade para seu servidor master. A Figura 4(b) mostra o nível de utilização do servidor com nós slaves. Quanto maior o número de recursos, menor o nível de utilização dos nós slaves. À medida que aumenta-se AD, o nível de utilização decai de forma sutil nos três casos. No entanto, esta queda somente inicia-se em AD = 3.0ms para SC = 8 e em AD = 1.5ms para SC = 16. Até estes pontos, o nível de utilização gira em torno de 82%, o que causa o comportamento do MRT explicado anteriormente.



(a) Utilização do Servidor Master



(b) Utilização do Servidor Slave

Figura 4. Nível de Utilização para os Servidores Master e Slaves

A Figura 5 apresenta a probabilidade de descarte de novas requisições. Para SC = 32 a probabilidade de descarte é igual a zero. Portanto, caso seja possível adquirir um servidor com 32 núcleos não haverá descarte independente do intervalo entre chegadas de requisições. Para SC = 8 e SC = 16 somente a partir de AD = 2.0ms e AD = 4.0 as probabilidades de descarte tendem a zero. Estes intervalos iniciais com descarte estão diretamente relacionados ao alto nível de utilização apresentados por ambos os servidores, impactando diretamente no tempo médio de resposta do sistema. Portanto, qualquer análise estocástica realizada com o modelo proposto deve observar as quatro métricas em questão para obter uma visão completa do comportamento do sistema. Também é possível identificar os limites de funcionamento do sistema, ou seja, até quantos trabalhos podem ser perdidos sem que comprometa a utilidade do sistema.

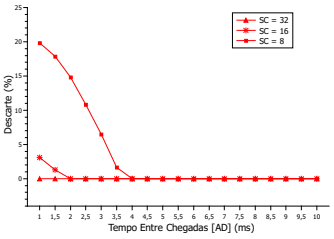


Figura 5. Descarte

4. Conclusão

Este artigo propôs um modelo SPN para representar e avaliar o desempenho de uma arquitetura MEC básica. O modelo permite estimar o Tempo Médio de Resposta (MRT) e o nível de utilização dos recursos na borda da rede. O avaliador pode configurar até 5 parâmetros de entrada no modelo, o que permite um alto nível de flexibilidade de

avaliação. Foi executada uma análise numérica baseando-se em dados reais de um artigo de referência que executou experimentos simplificados. A análise numérica permitiu observar o comportamento das quatro métricas (MRT, descarte, utilização master, e utilização slave) em função da variação do tempo entre chegadas de novas requisições, o que permitiu concluir que o tempo entre chegadas é um parâmetro de grande impacto sobre o sistema. Para valores muito pequenos (até 3.0ms) observou-se níveis de descarte, o que pode comprometer principalmente o tempo médio de resposta. Como trabalhos futuros, pretendemos estender o modelo para incluir requisitos de disponibilidade, mensurar gasto energético e explorar alocação entre múltiplas torres MEC.

Referências

Beck, M. T., Werner, M., Feld, S., and Schimper, S. (2014). Mobile edge computing: A taxonomy. In *Proc. of the Sixth International Conference on Advances in Future Internet*, pages 48–55. Citeseer.

Cau, E., Corici, M., Bellavista, P., Foschini, L., Carella, G., Edmonds, A., and Bohnert, T. M. (2016). Efficient exploitation of mobile edge computing for virtualized 5g in epc architectures. In *2016 4th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud)*, pages 100–109. IEEE.

Gopika Premsankar, M. d. F. and Taleb, T. (2018). Edge computing for the internet of things: A case study. 5:1275–1284.

Guangshun Li, J. W. and Junhua Wu, J. S. (2018). Data processing delay optimization in mobile edge computing. 2018.

Jain, R. (1990). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons.

Jararweh, Y., Doulat, A., Darabseh, A., Alsmirat, M., Al-Ayyoub, M., and Benkhelifa, E. (2016). Sdmec: Software defined system for mobile edge computing. In *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pages 88–93. IEEE.

Jung, H. (2016). Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020 white paper. Technical report, Technical report, Cisco Systems Inc.

Kitanov, S., Monteiro, E., and Janevski, T. (2016). 5g and the fog—survey of related technologies and research directions. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pages 1–6. IEEE.

Little, J. D. (1961). A proof for the queuing formula: L= λ w. *Operations research*, 9(3):383–387.

Orsini, G., Bade, D., and Lamersdorf, W. (2015). Computing at the mobile edge: Designing elastic android applications for computation offloading. In *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 112–119. IEEE.

Trinh, C. and Yao, L. (2017). Energy-aware mobile edge computing for low-latency visual data processing. pages 128–133.

Yuanzhe and Shangguang (2018). An energy-aware edge server placement algorithm in mobile edge computing. In *IEEE International Conference on Edge Computing (EDGE)*.